Figure 1

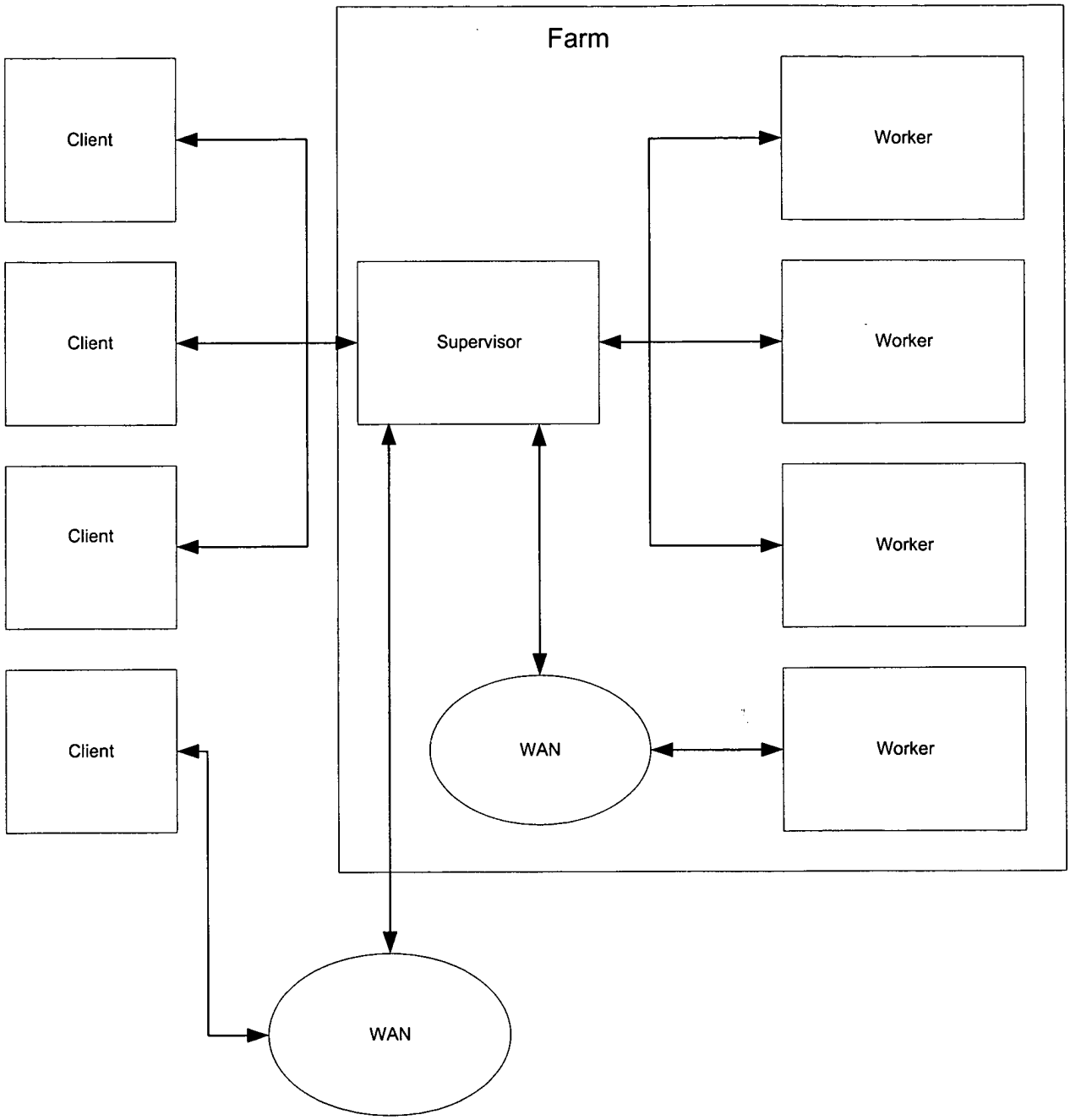
Farm
Client
Client
Client
Client
Supervisor
Worker
Worker
Worker
Worker
WAN
WAN

Figure 2

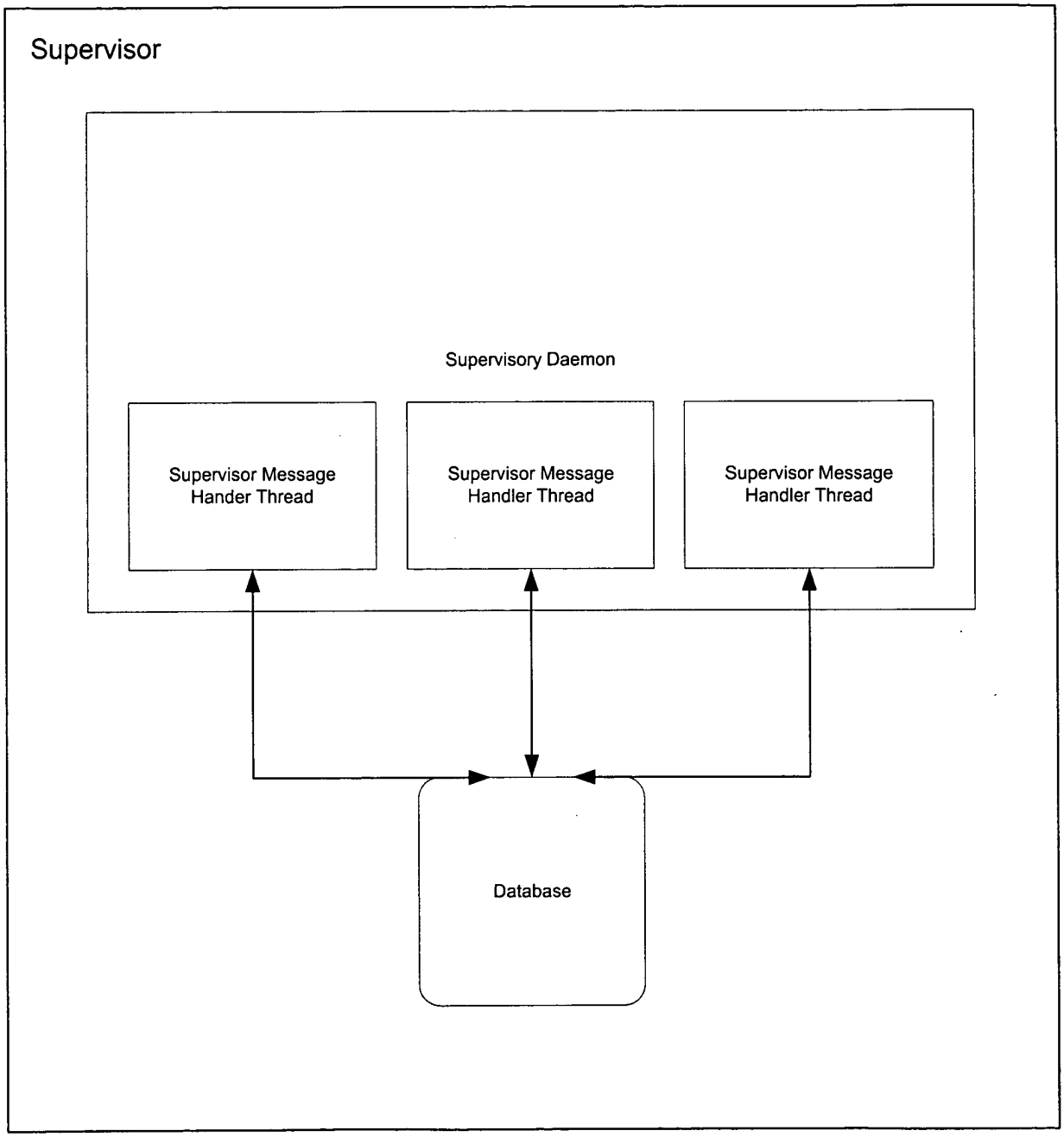
Supervisor
Supervisory Daemon
Supervisor Message Hander Thread
Supervisor Message Handler Thread
Supervisor Message Handler Thread
Database

Figure 3

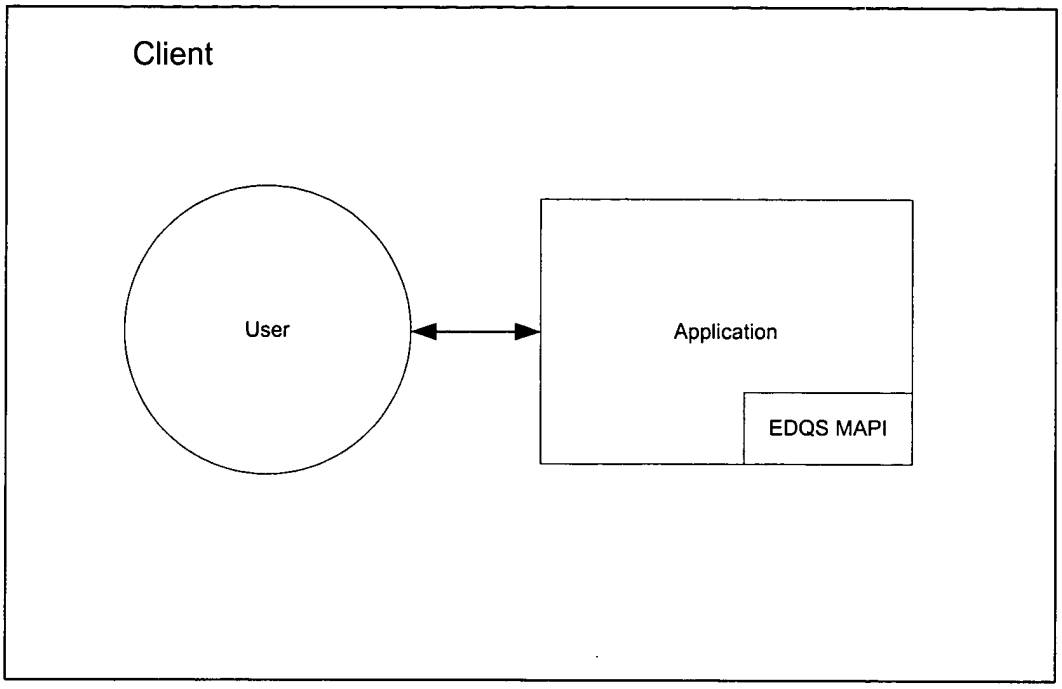

Figure 4

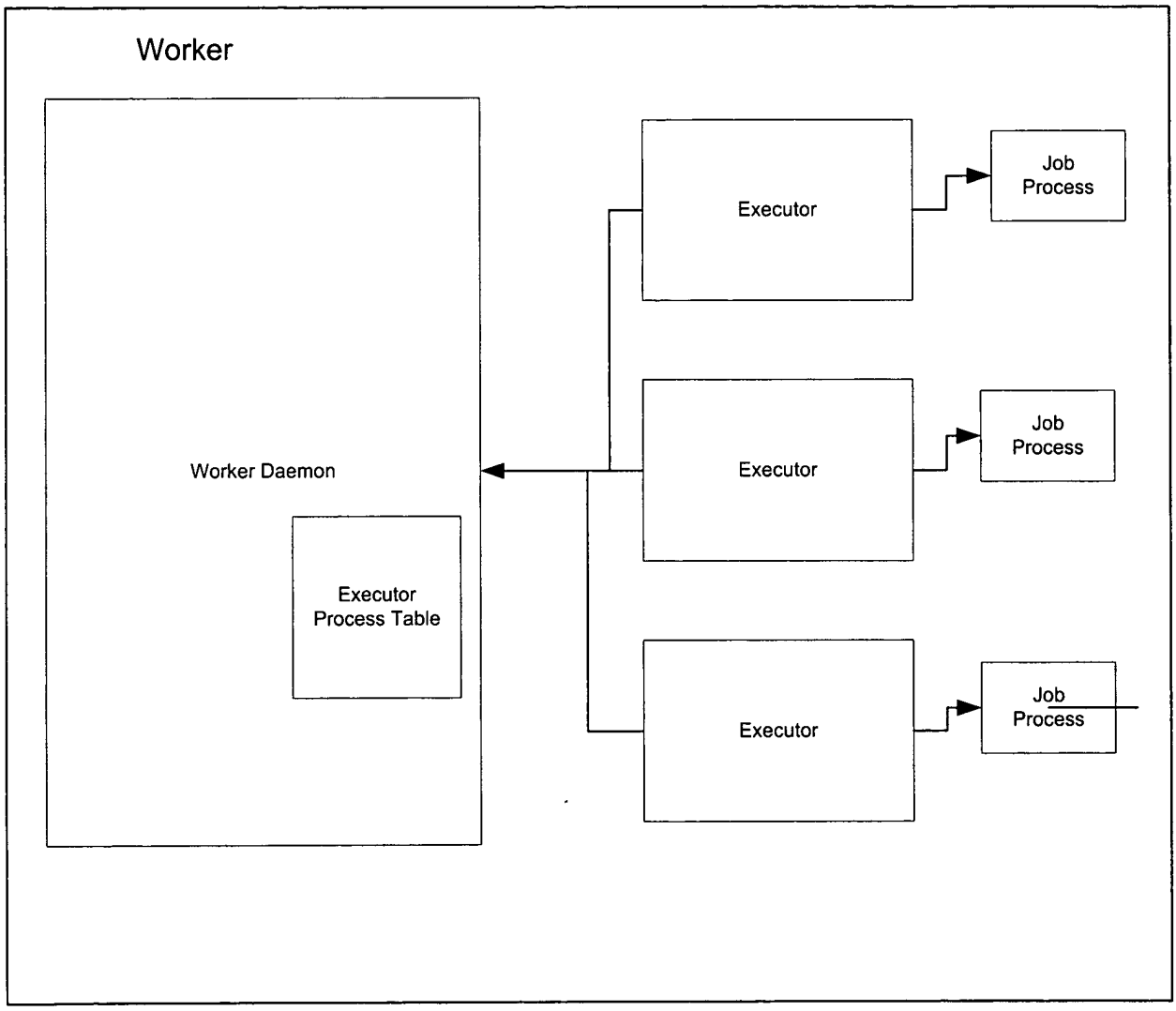
Worker
Worker Daemon
Executor Process Table
Executor
Job Process
Executor
Job Process
Executor
Job Process

Figure 5

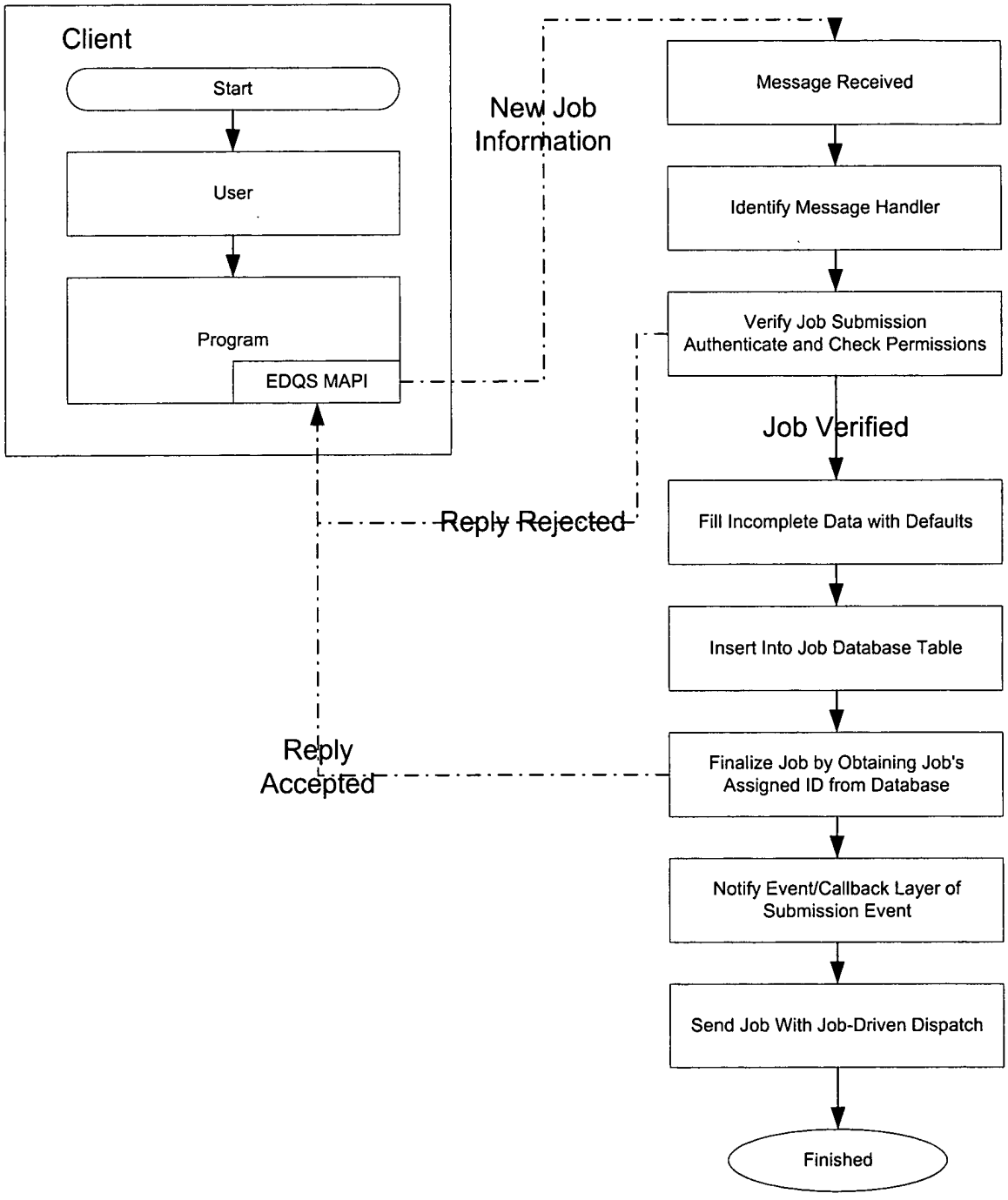
Client
Start
User
Program
EDQS MAPI
New Job Information
Message Received
Identify Message Handler
Verify Job Submission Authenticate and Check Permissions
Job Verified
Reply Rejected
Fill Incomplete Data with Defaults
Insert Into Job Database Table
Reply Accepted
Finalize Job by Obtaining Job's Assigned ID from Database
Notify Event/Callback Layer of Submission Event
Send Job With Job-Driven Dispatch
Finished

Figure 6

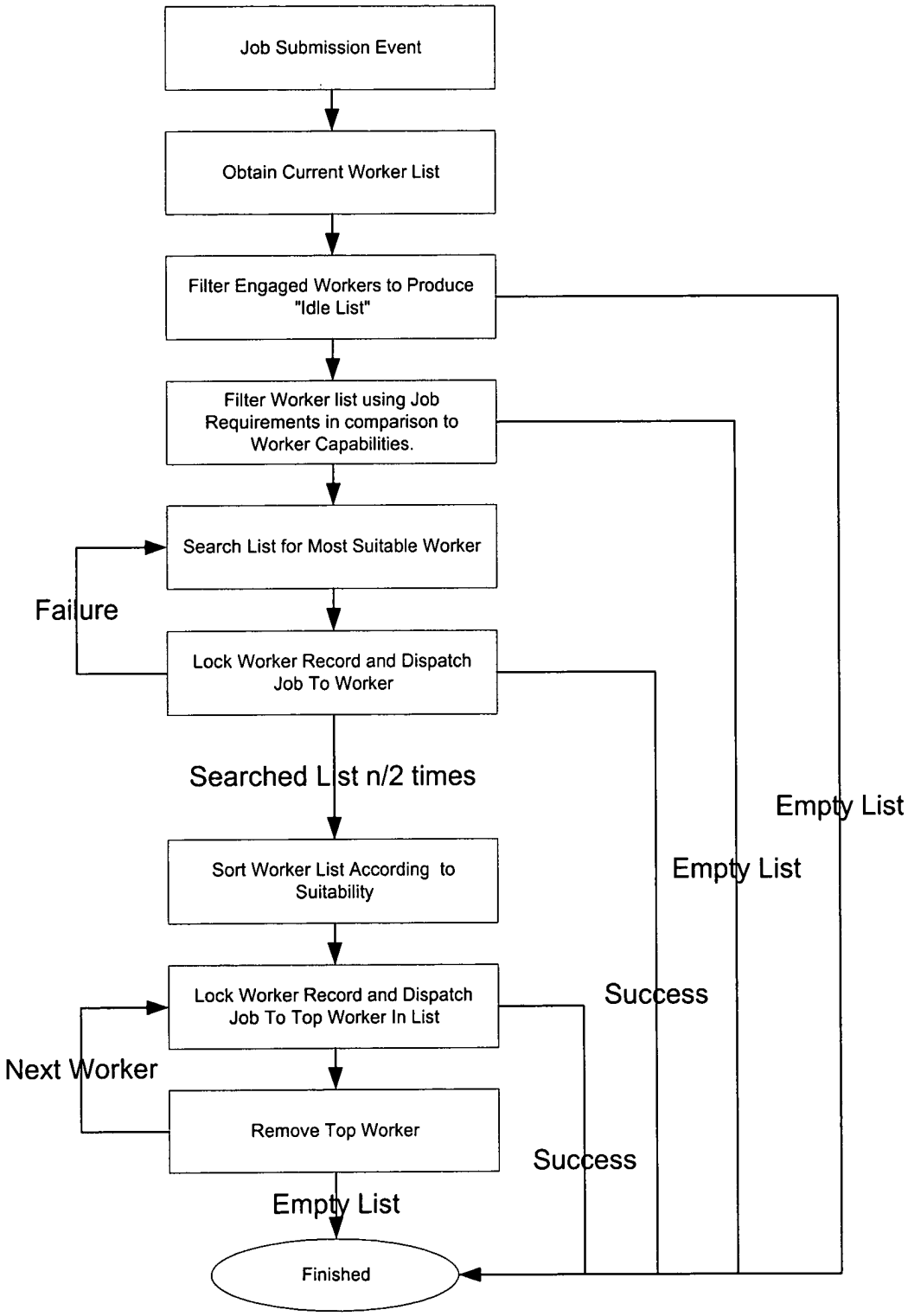
Job Submission Event
Obtain Current Worker List
Filter Engaged Workers to Produce "Idle List"
Filter Worker list using Job Requirements in comparison to Worker Capabilities.
Search List for Most Suitable Worker
Failure
Lock Worker Record and Dispatch Job To Worker
Searched List n/2 times
Sort Worker List According to Suitability
Lock Worker Record and Dispatch Job To Top Worker In List
Next Worker
Remove Top Worker
Empty List
Finished
Success
Success
Empty List
Empty List

Figure 7

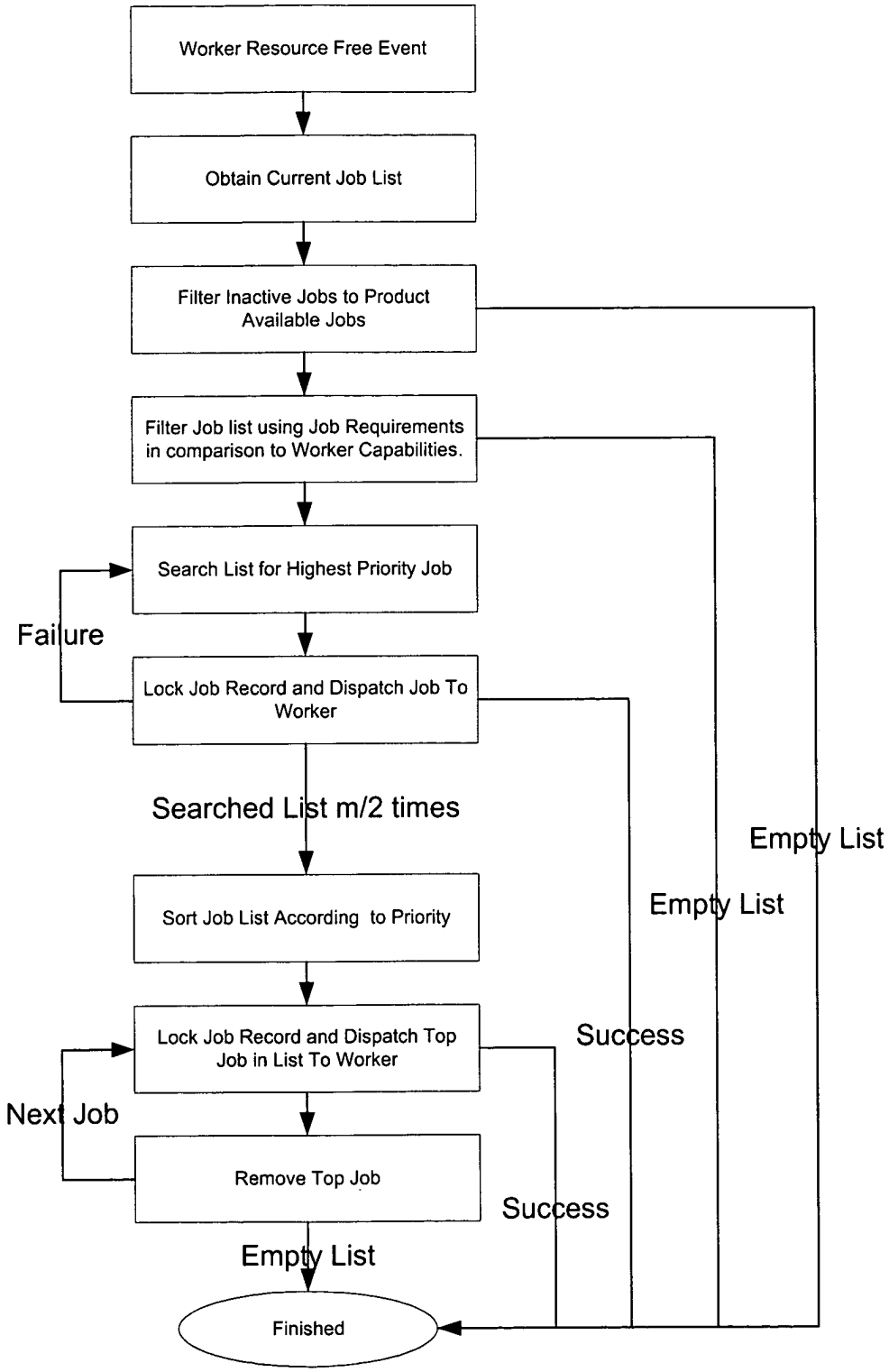
Worker Resource Free Event
Obtain Current Job List
Filter Inactive Jobs to Product Available Jobs
Filter Job list using Job Requirements in comparison to Worker Capabilities.
Search List for Highest Priority Job
Failure
Lock Job Record and Dispatch Job To Worker
Searched List m/2 times
Sort Job List According to Priority
Lock Job Record and Dispatch Top Job in List To Worker
Next Job
Remove Top Job
Empty List
Finished
Empty List
Empty List
Success
Success

Figure 8

```
SUBA: Initialize Job A's weight to 0
Starting from root node as reference node
If Job A home node = reference node
        Increment Job A's weight
Next reference node
Reset weight to 0
Goto SUBA

SUBB: Initialize Job B's Weight to 0
Starting from root node as reference node
If Job B home node = reference node
        Increment Job B's weight
Next reference node
Reset weight to 0
Goto SUBB

SUBC: Starting from root node as reference node
If Job A weight > Job B weight
Return Job A
Else If Job B weight >  Job A weight
        Return Job B
Else
        Return Equal
Next reference node
```

Figure 9A

Job A Home Node	Job B Home Node	Ref. node	Result
/	/project1	/	Equal
/project1	/	/	Equal
/project1	/project1	/	Equal
/	/project2	/	Equal
/project2	/	/	Equal
/project2	/project2	/	Equal
/project1	/project2	/	Equal
/project2	/project1	/	Equal
/	/project2/power	/	Equal
/project2/power	/	/	Equal
/project1	/project2/power	/	Equal
/project2/power	/project1	/	Equal
/project2	/project2/power	/	Equal
/project2/power	/project2	/	Equal
/project2/power	/project2/power	/	Equal
/	/	/project1	Equal
/	/project1	/project1	Job A < Job B
/project1	/	/project1	Job A > Job B
/project1	/project1	/project1	Equal
/	/project2	/project1	Equal
/project2	/	/project1	Equal
/project2	/project2	/project1	Equal
/project1	/project2	/project1	Job A > Job B
/project2	/project1	/project1	Job A < Job B
/	/project2/power	/project1	Equal
/project2/power	/	/project1	Equal
/project1	/project2/power	/project1	Job A > Job B
/project2/power	/project1	/project1	Job A < Job B
/project2	/project2/power	/project1	Equal
/project2/power	/project2	/project1	Equal
/project2/power	/project2/power	/project1	Equal
/	/	/project2	Equal
/	/project1	/project2	Equal
/project1	/	/project2	Equal
/project1	/project1	/project2	Equal
/	/project2	/project2	Job A < Job B
/project2	/	/project2	Job A > Job B
/project2	/project2	/project2	Equal
/project1	/project2	/project2	Job A > Job B
/project2	/project1	/project2	Job A < Job B

Figure 9B

/	/project2/power	/project2	Job A < Job B
/project2/power	/	/project2	Job A > Job B
/project1	/project2/power	/project2	Job A < Job B
/project2/power	/project1	/project2	Job A > Job B
/project2	/project2/power	/project2	Equal
/project2/power	/project2	/project2	Equal
/project2/power	/project2/power	/project2	Equal

/	/	/project2/power	Equal
/	/project1	/project2/power	Equal
/project1	/	/project2/power	Equal
/project1	/project1	/project2/power	Equal
/	/project2	/project2/power	Equal
/project2	/	/project2/power	Job A > Job B
/project2	/project2	/project2/power	Equal
/project1	/project2	/project2/power	Job A < Job B
/project2	/project1	/project2/power	Job A > Job B
/	/project2/power	/project2/power	Job A < Job B
/project2/power	/	/project2/power	Job A > Job B
/project1	/project2/power	/project2/power	Job A < Job B
/project2/power	/project1	/project2/power	Job A > Job B
/project2	/project2/power	/project2/power	Job A < Job B
/project2/power	/project2	/project2/power	Job A > Job B
/project2/power	/project2/power	/project2/power	Equal

Figure 10

```
SUBA: Initialize Job A's weight to 0
Starting from root node as reference node
If Job A home node = reference node
        Increment Job A's weight
Next reference node
Reset weight to 0
Goto SUBA

SUBB: Initialize Job B's weight to 0
Starting from root node as reference node
If Job B home node = reference node
        Increment Job B's weight
Next reference node
Reset weight to 0
Goto SUBB

SUBC: Starting from root node as reference node
If Job A weight > Job B weight
        Return Job A
Else If Job B weight >  Job A weight
        Return Job B
Else If Job A Priority > Job B Priority
Return Job A
Else If Job B Priority > Job A Priority
Return Job B
Else
        Return Equal
Next reference node
Goto SUBC
```

Figure 11A

#	Message Name	Description of Message Function	Fault Tolerance and Recovery if Message is Lost
1.	New Job	The initial message used by a user to submit a new job for processing.	Because New Job is normally initiated by the user, it is up to the user to retry in the event a Result Job (Message #2) is not received by the user.
2.	Result Job	Result Job is sent by a supervisor to a user interface to confirm that a job was submitted.	It is up to the user to retry in the event a Result Job (Message #2) is not received by the user. However, because of the nature of TCP/IP, it is unlikely this message would be lost if the New Job message was received by the supervisory daemon.
3.	Start Job	Start Job sent by supervisor to a worker to order the worker to start a processing job.	Loss of Start Job is typically resolved when the supervisor resends Start Job to the same node and obtains an Accept reply. If an Accept reply is not received, or is a Reject reply is received, the supervisor sends a Start Job message to a different suitable node.
4.	Accept/ Reject	Accept/Reject is used to convey the acceptance or rejection of a job by a worker. If the worker sends an acceptance as the contents of the message, the work undertakes processing and tracking the job. If the worker sends a rejection as the contents of the message, then the job is remains in one or more job queues.	Failure to receive Accept/Reject is not tolerated; the supervisor periodically resends a Start until an Accept/Reject is received or a time out or other decision point is reached. After a time-out, the supervisor normally sends a Remove Job to the worker that failed to respond, deletes that worker from the list of available workers, and sends Start Job to another suitable node.

Figure 11B

5.	Query Job	Query Job requests the status of a job being processed on a worker. The query occurs in order to transfer data from the worker to the querying process without using files.	Failure to receive a Job Detail in reply is handled by resending Query Job.
6.	Job Detail	Job Details responds to a Query Job, and provide information about the job and worker.	Loss of this message is handled by the querying process by resending Query Job.
7.	Job Status Query	Job Status Query queries the processing status of the job. It asks whether processing is continuing on the worker.	Failure to receive a Job Status Reply in reply is handled by resending Job Status Query until a Job Status Reply is received or a time out or other decision point is reached.
8.	Job Status Reply	Job Status Reply is sent by a worker to the querying process, usually on the supervisor. It is sent after the worker has recorded any new information contained in a Job Status Query in its internal cache.	Loss of Job Status Reply is tolerated and no resolution is required; the querying process periodically resends a Job Status Query, which should generate a subsequent Job Status Reply.
9.	Confirm	Confirm is sent to a worker by a supervisor in response to a Job Status Reply. It helps to ensure that the supervisor has correctly recorded the data.	Failure to receive a Reconfirm in reply is handled by resending Confirm until a Reconfirm is received or a time out or other decision point is reached.
10.	Reconfirm	Reconfirm is sent in response to Confirm.	Loss of Reconfirm is tolerated and no resolution is required; the sending process periodically resends a Confirm, which should generate a subsequent Reconfirm.

Figure 11C

11.	Worker Update	Immediately after a worker's capabilities have changed, the worker sends Worker Update to the supervisor.	Failure to receive an Update Confirm in reply is handled by resending Worker Update until an Update Confirm is received or a time out or other decision point is reached.
12.	Update Confirm	The supervisor responds to a Worker Update by sending the worker an Update Confirm.	Loss of Update Confirm is tolerated and no resolution is required; the sending process periodically resends a Worker Update, which should generate a subsequent Update Confirm.
13	Job Complete	Job Complete is sent by a worker to a supervisor upon completion of a processing job.	Failure to receive a Confirm Complete in reply is handled by resending Job Complete until a Confirm Complete is received or a time out or other decision point is reached.
14.	Confirm Complete	Confirm Complete is sent by a supervisor to a worker to confirm that the worker has completed a job.	Loss of Confirm Complete is tolerated and no resolution is required; the sending process periodically resends a Job Complete, which should generate a subsequent Confirm Complete.
15.	Remove Job	Remove Job is sent by a supervisor to a worker to terminate a job. It is used to interrupt job processing for a higher priority job, or because a related job requires that all jobs in a job group cease processing, or for other reasons.	Failure to receive a Confirm Remove in reply is handled by resending Remove Job until a Confirm Remove is received or a time out or other decision point is reached.

Figure 11D

16	Confirm Remove	Confirm Remove is sent by a worker to a supervisor to confirm that the relevant job has been removed.	Loss Confirm Remove is tolerated and no resolution is required; the sending process periodically resends a Remove Job, which should generate a subsequent Confirm Remove.
17	Worker Unavailable	Worker Unavailable is sent by a worker to a supervisor when before the worker goes off-line (becomes unavailable).	Failure to receive a Confirm Unavailable in reply is handled by resending Worker Unavailable until a Confirm Unavailable is received or the worker is forced off-line or fails.
18	Confirm Unavailable	Confirm Unavailable is sent by a supervisor to a worker to confirm that the worker is going off-line. The supervisor deletes the worker from the list of available workers.	Loss of the message is tolerated and no resolution is required; the worker goes off-line or fails.

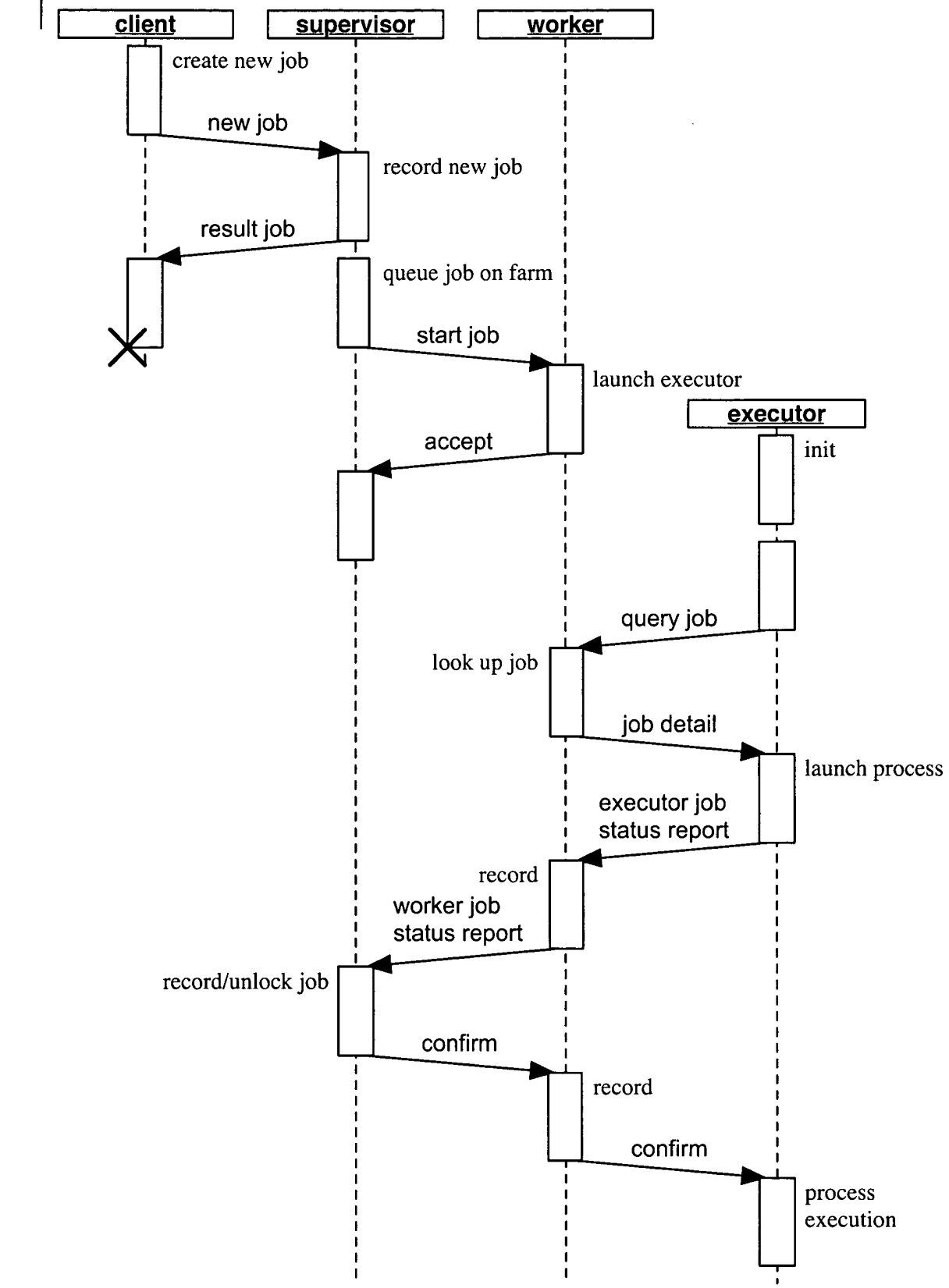
client
supervisor
worker
create new job
new job
record new job
result job
queue job on farm
start job
launch executor
executor
accept
init
query job
look up job
job detail
launch process
executor job
status report
record
worker job
status report
record/unlock job
confirm
record
confirm
process
execution

Figure 12A

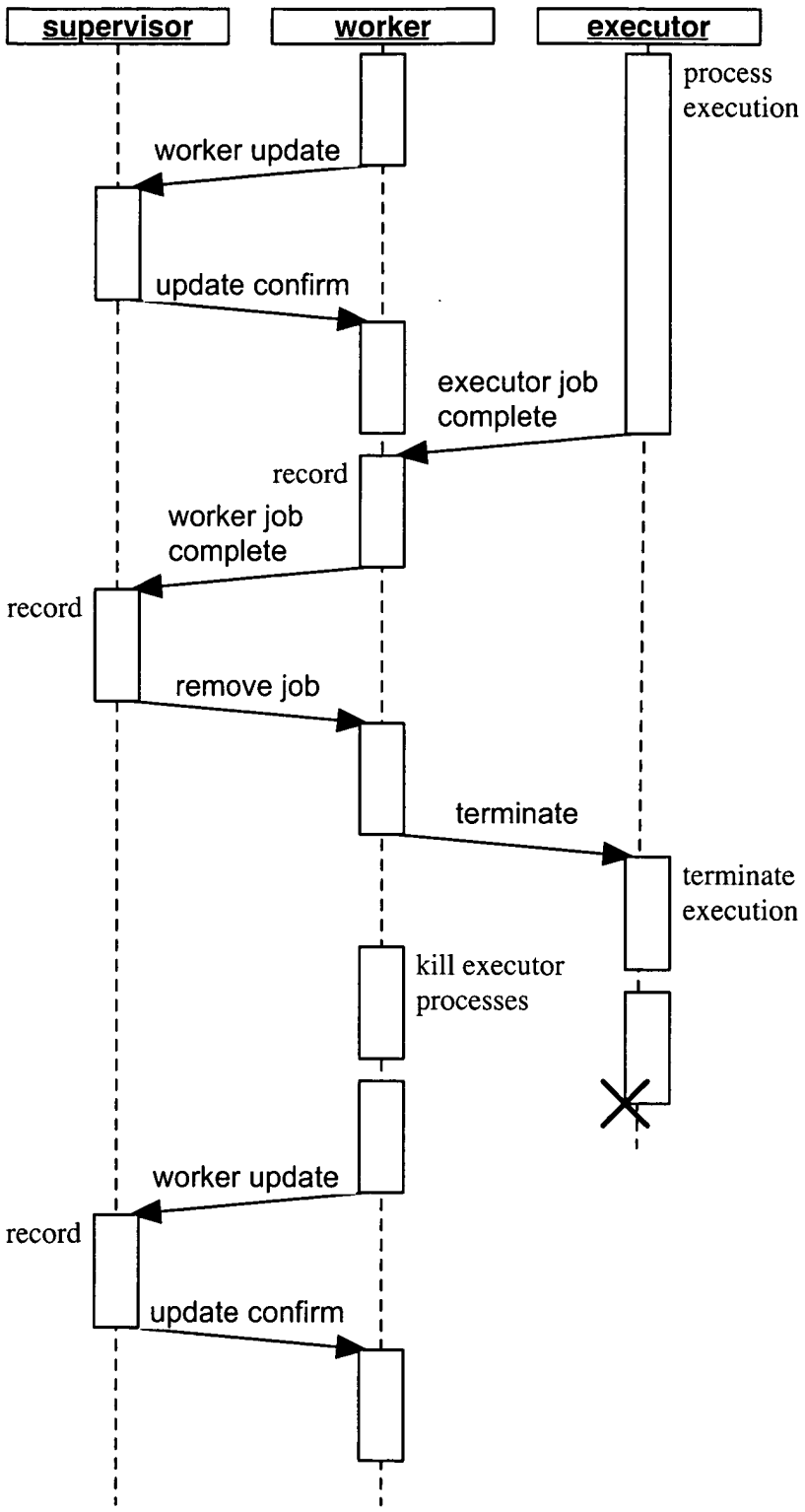
supervisor
worker
executor
process execution
worker update
update confirm
executor job complete
record
worker job complete
record
remove job
terminate
terminate execution
kill executor processes
worker update
record
update confirm

Figure 12B

Figure 13

Job Status

Status	Value
NONE	0x0
COMPLETE	0x10
FAILED	0x20
KILLED	0x30
BLOCKED	0x110
WAITING	0x120
SUSPENDED	0x130
PENDING	0x140
RUNNING	0x150

Figure 14

Evtypes

evtype	Value
JOB	0
SUBJOB	1
WORK	2
CUSTOM	3
NONE	4
TIME	8
REPEAT	16
HOST	32
GLOBAL	128
GLOBALTIME	136
GLOBALREPEAT	152

Figure 15A

Typical Events

memorylimit-job-self
timelimit-job-self
submit-job-self
failed-job-self
complete-job-self
killed-job-self
blocked-job-self
running-job-self
removed-job-self
suspended-job-self
waiting-job-self
assigned-job-self
done-job-self
modified-job-self

memorylimit-subjob-self-<subjob id>
timelimit-subjob-self-<subjob id>
submit-subjob-self-<subjob id>
failed-subjob-self-<subjob id>
complete-subjob-self-<subjob id>
killed-subjob-self-<subjob id>
blocked-subjob-self-<subjob id>
running-subjob-self-<subjob id>
removed-subjob-self-<subjob id>
suspended-subjob-self-<subjob id>
waiting-subjob-self-<subjob id>
assigned-subjob-self-<subjob id>
done-subjob-self-<subjob id>
modified-subjob-self-<subjob id>

memorylimit-work-self-<agenda id>
timelimit-work-self-<agenda id>
submit-work-self-<agenda id>
failed-work-self-<agenda id>
complete-work-self-<agenda id>
killed-work-self-<agenda id>
blocked-work-self-<agenda id>
running-work-self-<agenda id>
removed-work-self-<agenda id>
suspended-work-self-<agenda id>
waiting-work-self-<agenda id>

Figure 15B

assigned-work-self-<agenda id>
done-work-self-<agenda id>
modified-work-self-<agenda id>

memorylimit-job-parent
timelimit-job-parent
submit-job-parent
failed-job-parent
complete-job-parent
killed-job-parent
blocked-job-parent
running-job-parent
removed-job-parent
suspended-job-parent
waiting-job-parent
assigned-job-parent
done-job-parent
modified-job-parent

memorylimit-subjob-parent-<subjob id>
timelimit-subjob-parent-<subjob id>
submit-subjob-parent-<subjob id>
failed-subjob-parent-<subjob id>
complete-subjob-parent-<subjob id>
killed-subjob-parent-<subjob id>
blocked-subjob-parent-<subjob id>
running-subjob-parent-<subjob id>
removed-subjob-parent-<subjob id>
suspended-subjob-parent-<subjob id>
waiting-subjob-parent-<subjob id>
assigned-subjob-parent-<subjob id>
done-subjob-parent-<subjob id>
modified-subjob-parent-<subjob id>

memorylimit-work-parent-<agenda id>
timelimit-work-parent-<agenda id>
submit-work-parent-<agenda id>
failed-work-parent-<agenda id>
complete-work-parent-<agenda id>
killed-work-parent-<agenda id>
blocked-work-parent-<agenda id>

Figure 15C

running-work-parent-<agenda id>
removed-work-parent-<agenda id>
suspended-work-parent-<agenda id>
waiting-work-parent-<agenda id>
assigned-work-parent-<agenda id>
done-work-parent-<agenda id>
modified-work-parent-<agenda id>

memorylimit-job-<label>
timelimit-job-<label>
submit-job-<label>
failed-job-<label>
complete-job-<label>
killed-job-<label>
blocked-job-<label>
running-job-<label>
removed-job-<label>
suspended-job-<label>
waiting-job-<label>
assigned-job-<label>
done-job-<label>
modified-job-<label>

memorylimit-subjob-<label>-<subjob id>
timelimit-subjob-<label>-<subjob id>
submit-subjob-<label>-<subjob id>
failed-subjob-<label>-<subjob id>
complete-subjob-<label>-<subjob id>
killed-subjob-<label>-<subjob id>
blocked-subjob-<label>-<subjob id>
running-subjob-<label>-<subjob id>
removed-subjob-<label>-<subjob id>
suspended-subjob-<label>-<subjob id>
waiting-subjob-<label>-<subjob id>
assigned-subjob-<label>-<subjob id>
done-subjob-<label>-<subjob id>
modified-subjob-<label>-<subjob id>

memorylimit-work-<label>-<agenda id>
timelimit-work-<label>-<agenda id>
submit-work-<label>-<agenda id>

Figure 15D

failed-work-<label>-<agenda id>
complete-work-<label>-<agenda id>
killed-work-<label>-<agenda id>
blocked-work-<label>-<agenda id>
running-work-<label>-<agenda id>
removed-work-<label>-<agenda id>
suspended-work-<label>-<agenda id>
waiting-work-<label>-<agenda id>
assigned-work-<label>-<agenda id>
done-work-<label>-<agenda id>
modified-work-<label>-<agenda id>

memorylimit-job-<job_id>
timelimit-job-<job_id>
submit-job-<job_id>
failed-job-<job_id>
complete-job-<job_id>
killed-job-<job_id>
blocked-job-<job_id>
running-job-<job_id>
removed-job-<job_id>
suspended-job-<job_id>
waiting-job-<job_id>
assigned-job-<job_id>
done-job-<job_id>
modified-job-<job_id>

memorylimit-subjob-<job_id>-<subjob id>
timelimit-subjob-<job_id>-<subjob id>
submit-subjob-<job_id>-<subjob id>
failed-subjob-<job_id>-<subjob id>
complete-subjob-<job_id>-<subjob id>
killed-subjob-<job_id>-<subjob id>
blocked-subjob-<job_id>-<subjob id>
running-subjob-<job_id>-<subjob id>
removed-subjob-<job_id>-<subjob id>
suspended-subjob-<job_id>-<subjob id>
waiting-subjob-<job_id>-<subjob id>
assigned-subjob-<job_id>-<subjob id>
done-subjob-<job_id>-<subjob id>
modified-subjob-<job_id>-<subjob id>

Figure 15E

memorylimit-work-<job_id>-<agenda id>
timelimit-work-<job_id>-<agenda id>
submit-work-<job_id>-<agenda id>
failed-work-<job_id>-<agenda id>
complete-work-<job_id>-<agenda id>
killed-work-<job_id>-<agenda id>
blocked-work-<job_id>-<agenda id>
running-work-<job_id>-<agenda id>
removed-work-<job_id>-<agenda id>
suspended-work-<job_id>-<agenda id>
waiting-work-<job_id>-<agenda id>
assigned-work-<job_id>-<agenda id>
done-work-<job_id>-<agenda id>
modified-work-<job_id>-<agenda id>

Figure 16

Agenda Table

Field	Type	Description
name	TEXT	Name of agenda item
INDEX	(name(16))	Index field based on the name
status	INTEGER	Status of agenda item
host	CHAR(255)	Work item execution host
data	TEXT	Pointer to agenda item data structure
subid	INTEGER	Agenda item subjob ID
address	INTEGER	Worker Host IP Address
lastupdate	INTEGER	Time of last update

Figure 17

Callback Helper Table

Field	Type	Description
uq	INTEGER PRIMARY KEY AUTO_INCREMENT	Unique ID of event
pid	INTEGER	Parent ID associated with event
pgrp	INTEGER	Process group ID associated with event
category	INTEGER	Event category
jobid	INTEGER	Job ID associated with event
subid	INTEGER	Subjob ID associated with event
workid	TEXT	Worker ID associated with event
start	INTEGER	Start Time to Execute
inter	INTEGER	Interval Time to Execute
extra	TEXT	Extra data to describe event
name	TEXT	Event's name
flag	INTEGER	
callback	INTEGER	ID of associated callback
label	TEXT	Event label
lastupdate	INTEGER	Time of last update

Figure 18

Callback Table

Field	Type	Description
id	INTEGER PRIMARY KEY AUTO_INCREMENT	Callback ID Number
language	INTEGER	Language binding for callback code execution
code	TEXT	Callback code to be executed if triggers evaluate to TRUE
count	INTEGER	Total times callback was executed
max	INTEGER	Maximum times callback will execute
triggers	TEXT	Expression combining list of events needed to activate callback code. Expression must evaluate to TRUE before callback code can be executed.
ready	TEXT	Event check list encoded as text
user	TEXT	User name callback registered to
lastupdate	INTEGER	Time of last update

Figure 19

Subjob Table

Field	Type	Description
id	INTEGER	ID number of subjob
status	INTEGER	Current status of subjob
host	TEXT	Name of host executing subjob
address	INTEGER	IP address of host executing subjob
reserve	INTEGER	(0 by default) Reservation Lock
lastupdate	INTEGER	Time of last update
data	TEXT	Subjob Package data

Figure 20

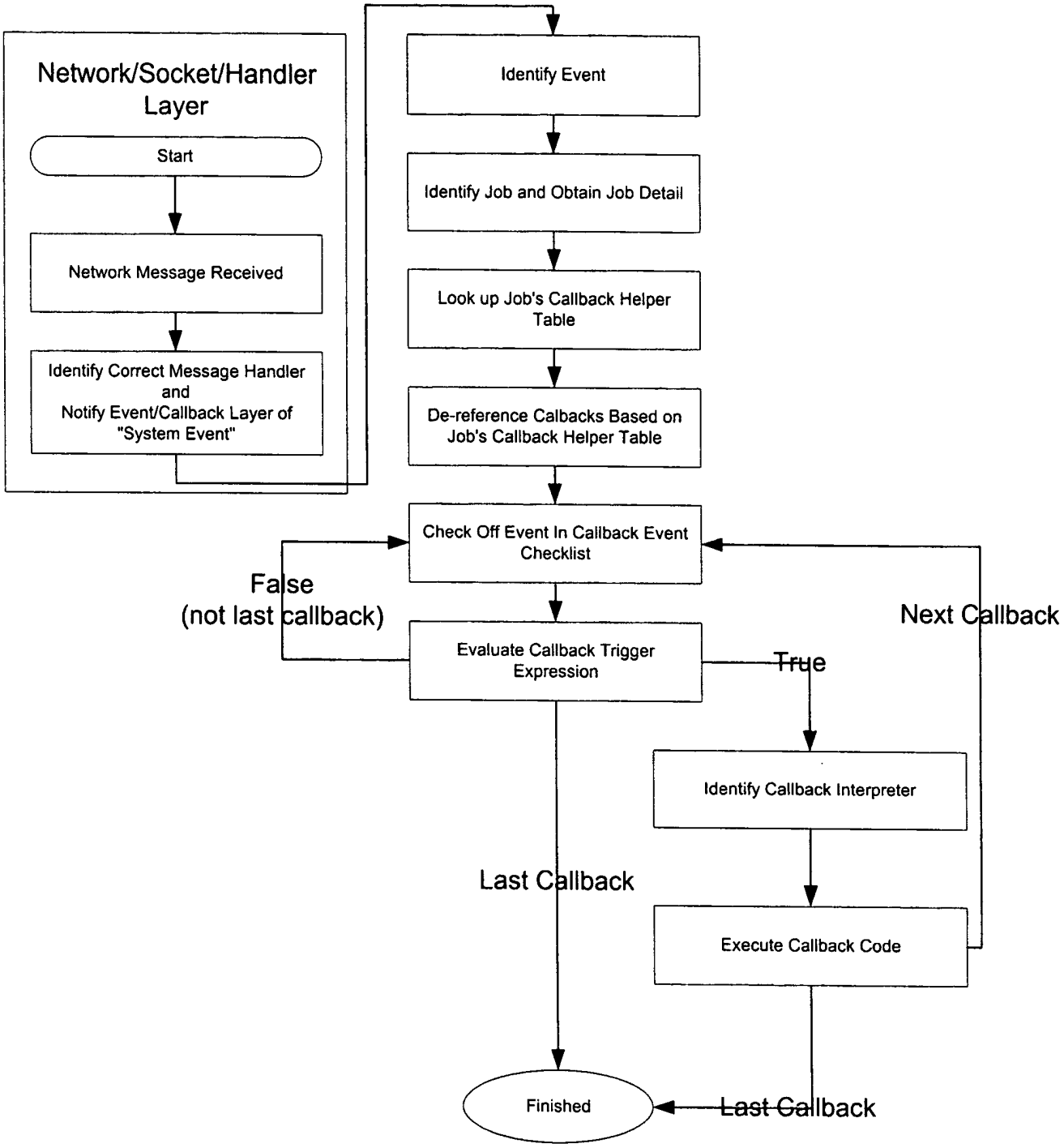
Network/Socket/Handler Layer
Start
Network Message Received
Identify Correct Message Handler and Notify Event/Callback Layer of "System Event"
Identify Event
Identify Job and Obtain Job Detail
Look up Job's Callback Helper Table
De-reference Calbacks Based on Job's Callback Helper Table
Check Off Event In Callback Event Checklist
False (not last callback)
Evaluate Callback Trigger Expression
True
Next Callback
Identify Callback Interpreter
Last Callback
Execute Callback Code
Finished
Last Callback

Figure 21A

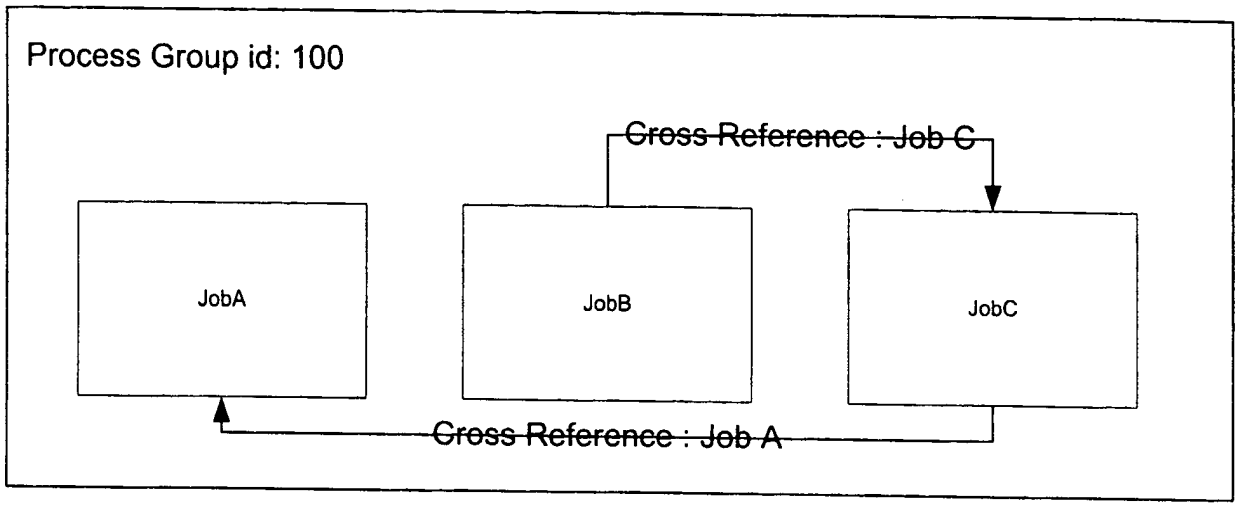

Figure 21B

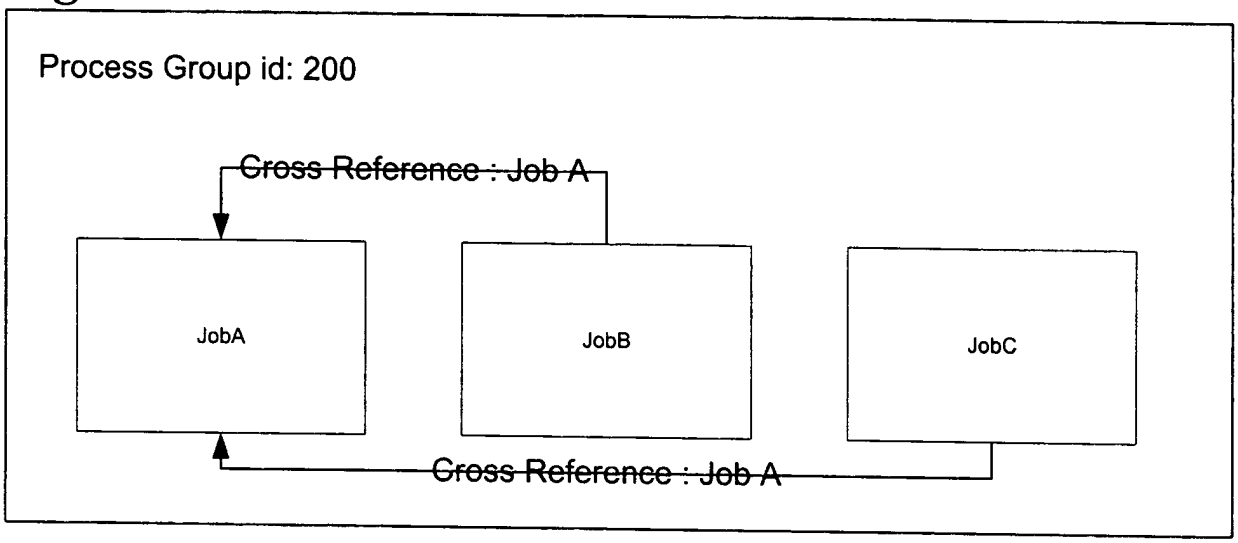
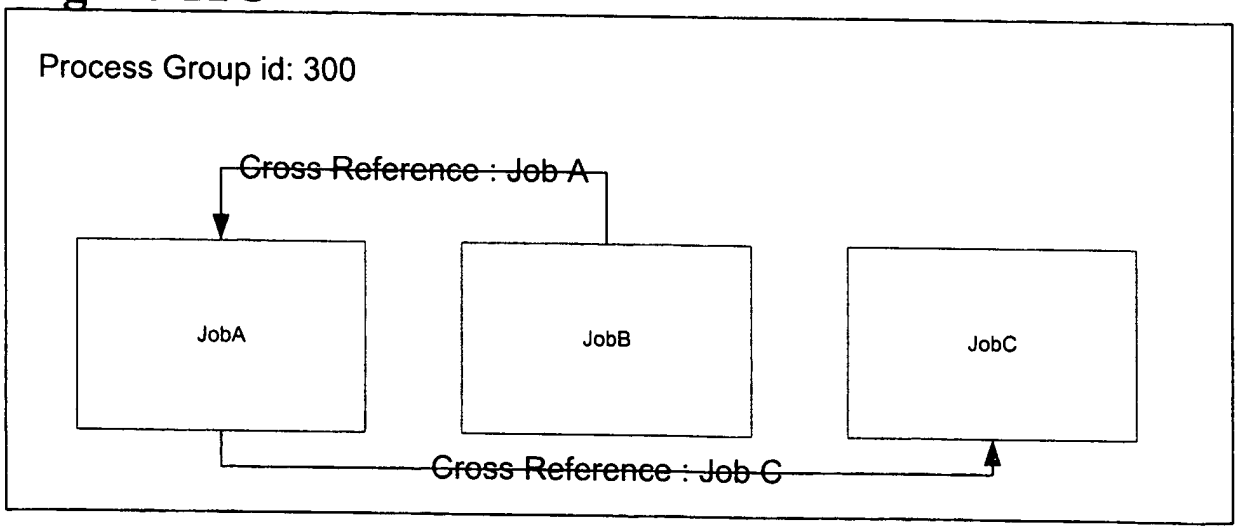
Figure 21C